All Contests > СДА Домашно 4 > Delete a Node

Delete a Node



Problem

Submissions

Leaderboard

Discussions

This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

Delete the node at a given position in a linked list and return a reference to the head node. The head is at position 0. The list may be empty after you delete the node. In that case, return a null value.

Example

$$\textit{llist} = 0 \rightarrow 1 \rightarrow 2 \rightarrow 3$$

$$position = 2$$

After removing the node at position **2**, $llist' = 0 \rightarrow 1 \rightarrow 3$.

Function Description

Complete the *deleteNode* function in the editor below.

deleteNode has the following parameters:

- SinglyLinkedListNode pointer llist: a reference to the head node in the list
- int position: the position of the node to remove

Returns

- SinglyLinkedListNode pointer: a reference to the head of the modified list

Input Format

The first line of input contains an integer n, the number of elements in the linked list.

Each of the next n lines contains an integer, the node data values in order.

The last line contains an integer, **position**, the position of the node to delete.

Constraints

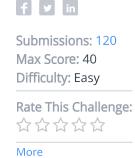
- $1 \le n \le 1000$
- $1 \leq list[i] \leq 1000$, where list[i] is the i^{th} element of the linked list.

Sample Input

- 8 20
- 6
- 2
- 19
- 7 4
- 15
- 9
- 3

Sample Output

The original list is $20 \rightarrow 6 \rightarrow 2 \rightarrow 19 \rightarrow 7 \rightarrow 4 \rightarrow 15 \rightarrow 9$. After deleting the node at position 3, the list is $20 \rightarrow 6 \rightarrow 2 \rightarrow 7 \rightarrow 4 \rightarrow 15 \rightarrow 9$.



```
C++14
                                                                                                 X | 🌣
 1 ▶#include ↔
 2
 3
   using namespace std;
 4
 5 ▼class SinglyLinkedListNode {
 6
        public:
 7
            int data;
 8
            SinglyLinkedListNode *next;
 9
10
            SinglyLinkedListNode(int node_data) {
11
                this->data = node_data;
12
                this->next = nullptr;
13
            }
14
   };
15
16 ▼class SinglyLinkedList {
17
        public:
18
            SinglyLinkedListNode *head;
19
            SinglyLinkedListNode *tail;
20
21
            SinglyLinkedList() {
22
                this->head = nullptr;
23
                this->tail = nullptr;
24
            }
25
            void insert_node(int node_data) {
26 ▼
                SinglyLinkedListNode* node = new SinglyLinkedListNode(node_data);
27
28
29 1
                if (!this->head) {
30
                    this->head = node;
31 🔻
                } else {
32
                    this->tail->next = node;
                }
34
35
                this->tail = node;
            }
36
37
   };
38
  void print_singly_linked_list(SinglyLinkedListNode* node, string sep, ofstream& fout) {
40
        while (node) {
            fout << node->data;
41
42
43
            node = node->next;
44
            if (node) {
45
                fout << sep;
46
            }
47
48
        }
49
   }
50
51 void free_singly_linked_list(SinglyLinkedListNode* node) {
52 ▼
        while (node) {
```

```
53
            SinglyLinkedListNode* temp = node;
54
            node = node->next;
55
56
            free(temp);
57
        }
58 }
59 ▼/*
    * Complete the 'deleteNode' function below.
60
61
     * The function is expected to return an INTEGER_SINGLY_LINKED_LIST.
62
     * The function accepts following parameters:
63

    INTEGER_SINGLY_LINKED_LIST llist

64
65
        2. INTEGER position
66
67
68 ▼/*
69
    * For your reference:
70
71
     * SinglyLinkedListNode {
72
           int data;
73
           SinglyLinkedListNode* next;
74
     * };
75
76
77
78 \singlyLinkedListNode* deleteNode(SinglyLinkedListNode* llist, int position) {
79
80
   }
 81 int main()
 82 ▼{
         ofstream fout(getenv("OUTPUT_PATH"));
 83
 84
 85
         SinglyLinkedList* llist = new SinglyLinkedList();
 86
 87
         int llist_count;
         cin >> llist_count;
 88
         cin.ignore(numeric_limits<streamsize>::max(), '\n');
 89
 90
 91
         for (int i = 0; i < llist_count; i++) {</pre>
 92
             int llist_item;
 93
             cin >> llist_item;
             cin.ignore(numeric_limits<streamsize>::max(), '\n');
 94
 95
             llist->insert_node(llist_item);
 96
 97
         }
 98
 99
         int position;
100
         cin >> position;
         cin.ignore(numeric_limits<streamsize>::max(), '\n');
101
102
         SinglyLinkedListNode* llist1 = deleteNode(llist->head, position);
103
104
105
         print_singly_linked_list(llist1, " ", fout);
         fout << "\n";
106
107
         free_singly_linked_list(llist1);
108
109
110
         fout.close();
111
         return 0;
112
    }
113
114
```

Line: 24 Col: 1